- -> how you access and change specific rows and columns
- ***-> a = np.array <- he has defined an array which is 2x7***

```
In [9]: a = np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
        print(a)

        [[ 1  2  3  4  5  6  7]
         [ 8  9 10 11 12 13 14]]
```

  - -> you can return this with a.shape
  - -> a[1,5] <- to return the element at this row and this column (13)
  - -> negatives also work for this -> for example a[1,-2]
- -> you can also use comments #
- -> to extract a specific row or column
  - -> a[0,:] <- this returns everything in the row
  - -> a[:,2] <- for everything in a column

- ***-> slicing***
  - -> start:stop:step
  - -> a[0,1:6:2] <- give me the first row, start at the first element, end at the 6th element and increase in increments (steps) of 2
  - -> this is an example of slicing
  - -> you can also use -2 <- to go backwards
  - -> to access elements
  - -> you can also change elements
  - -> you can do the same thing for an entire series of numbers (a column)
  - -> a[:2] = [1,2] <- you can set certain elements of the array equal to others

- ***-> he's defined and printed another array***

```
In [25]: b = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
         print(b)

         [[[1 2]
           [3 4]]

          [[5 6]
           [7 8]]]
```

  - -> the b array
  - -> then extracting elements from it
  - -> it's looking at the rows and the columns which we want
  - -> you can do the same thing with colons
    - -> start:stop:step with colons
    - -> for example b[:,1,:]
  - -> this is indexing
  - -> you can also do advanced indexing

```
Out[30]: 4

In [32]: # replace
         b[:,1,:] = [[9,9],[8,8]]
```

- ***-> question***

What code would change the values in the 3rd column of both of the following Numpy arrays to 20?

a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

Output:
[[1 2 20 4 5]
[6 7 20 9 10]]

a[:, 3] = 20

a[2, :] = 20

a[:, 2] = 20 <- This one

a[1, 2] = 20